## AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application:

## Listing of Claims:

1      1.      (Currently amended) A method for reducing the overhead involved
2   in executing native code methods in an application running on a virtual machine,
3   comprising:
4          selecting a call to any native code method to be optimized within the
5   virtual machine;
6          decompiling at least part of the selected native code method into an
7   intermediate representation;
8          obtaining an intermediate representation associated with the application
9   running on the virtual machine which interacts with the native code method;
10          integrating the intermediate representation for the native code method into
11   the intermediate representation associated with the application running on the
12   virtual machine to form an integrated intermediate representation; and
13          generating native code from the integrated intermediate representation,
14   wherein the native code generation process optimizes interactions between the
15   application running on the virtual machine and the native code method, wherein
16   optimizing the interactions involves optimizing calls from the application to the
17   native code method by using additional information from the integrated
18   intermediate representation to reduce the number of indirect calls and indirect
19   references associated with the calls, wherein optimizing the interactions involves
20   using contextual information from within the integrated intermediate
21   representation that is generated from the native code method as well as the

2

22 ~~application in order to optimize calls to the native code method by the application~~
23 ~~within the integrated intermediate representation.~~

1      2.     (Original) The method of claim 1, wherein selecting the call to the
2 native code method involves selecting the call based upon at least one of:
3          the execution frequency of the call; and
4          the overhead involved in performing the call to the native code method as
5 compared against the amount of work performed by the native code method.

1      3     (Canceled).

1      4.     (Original) The method of claim 1, wherein optimizing interactions
2 between the application running on the virtual machine and the native code
3 method involves optimizing callbacks by the native code method into the virtual
4 machine.

1      5.     (Original) The method of claim 4, wherein optimizing callbacks by
2 the native code method into the virtual machine involves optimizing callbacks
3 that access heap objects within the virtual machine.

1      6.     (Previously presented) The method of claim 4,
2          wherein the virtual machine is a platform-independent virtual machine;
3 and
4          wherein combining the intermediate representation for the native code
5 method with the intermediate representation associated with the application
6 running on the virtual machine involves integrating calls provided by an interface
7 for accessing native code into the native code method.

3

1       7.     (Original) The method of claim 1, wherein obtaining the
2    intermediate representation associated with the application running on the virtual
3    machine involves recompiling a corresponding portion of the application.

1       8.     (Original) The method of claim 1, wherein obtaining the
2    intermediate representation associated the application running on the virtual
3    machine involves accessing a previously generated intermediate representation
4    associated with the application running on the virtual machine.

1       9.     (Original) The method of claim 1, wherein prior to decompiling
2    the native code method, the method further comprises setting up a context for the
3    decompilation by:
4            determining a signature of the call to the native code method; and
5            determining a mapping from arguments of the call to corresponding
6    locations in a native application binary interface (ABI).

1     10.    (Currently amended) A computer-readable storage device storing
2    instructions that when executed by a computer cause the computer to perform a
3    method for reducing the overhead involved in executing native code methods in
4    an application running on a virtual machine, the method comprising:
5            selecting a call to any native code method to be optimized within the
6    virtual machine;
7            decompiling at least part of the selected native code method into an
8    intermediate representation;
9            obtaining an intermediate representation associated with the application
10   running on the virtual machine which interacts with the native code method;

11        integrating the intermediate representation for the native code method into

12    the intermediate representation associated with the application running on the

13    virtual machine to form an integrated intermediate representation; and

14        generating native code from the integrated intermediate representation,

15    wherein the native code generation process optimizes interactions between the

16    application running on the virtual machine and the native code method, wherein

17    optimizing the interactions involves optimizing calls from the application to the

18    native code method by using additional information from the integrated

19    intermediate representation to reduce the number of indirect calls and indirect

20    references associated with the calls~~wherein optimizing the interactions involves~~

21    ~~using contextual information from within the integrated intermediate~~

22    ~~representation that is generated from the native code method as well as the~~

23    ~~application in order to optimize calls to the native code method by the application~~

24    ~~within the integrated intermediate representation.~~

1      11.    (Previously presented) The computer-readable storage device of

2    claim 10, wherein selecting the call to the native code method involves selecting

3    the call based upon at least one of:

4        the execution frequency of the call; and

5        the overhead involved in performing the call to the native code method as

6    compared against the amount of work performed by the native code method.

1      12    (Cancelled).

1      13.    (Previously presented) The computer-readable storage device of

2    claim 10, wherein optimizing interactions between the application running on the

3    virtual machine and the native code method involves optimizing callbacks by the

4    native code method into the virtual machine.

<div align="center">5</div>

1    14.    (Previously presented) The computer-readable storage device of
2    claim 13, wherein optimizing callbacks by the native code method into the virtual
3    machine involves optimizing callbacks that access heap objects within the virtual
4    machine.

1    15.    (Previously presented) The computer-readable storage device of
2    claim 13,
3         wherein the virtual machine is a platform-independent virtual machine;
4    and
5         wherein combining the intermediate representation for the native code
6    method with the intermediate representation associated with the application
7    running on the virtual machine involves integrating calls provided by an interface
8    for accessing native code into the native code method.

1    16.    (Previously presented) The computer-readable storage device of
2    claim 10, wherein obtaining the intermediate representation associated with the
3    application running on the virtual machine involves recompiling a corresponding
4    portion of the application.

1    17.    (Previously presented) The computer-readable storage device of
2    claim 10, wherein obtaining the intermediate representation associated with the
3    application running on the virtual machine involves accessing a previously
4    generated intermediate representation associated with the application running on
5    the virtual machine.

6

1      18.    (Previously presented) The computer-readable storage device of
2  claim 10, wherein prior to decompiling the native code method, the method
3  further comprises setting up a context for the decompilation by:
4        determining a signature of the call to the native code method; and
5        determining a mapping from arguments of the call to corresponding
6  locations in a native application binary interface (ABI).

1      19-27. (Cancelled)

1      28.    (Currently amended) A method for reducing the overhead involved
2  in executing native code methods in an application running on a virtual machine,
3  comprising:
4        deciding to optimize a callback by any native code method into the virtual
5  machine;
6        decompiling at least part of the selected native code method into an
7  intermediate representation;
8        obtaining an intermediate representation associated with the application
9  running on the virtual machine which interacts with the native code method;
10      integrating the intermediate representation for the native code method into
11  the intermediate representation associated with the application running on the
12  virtual machine to form an integrated intermediate representation; and
13      generating native code from the integrated intermediate representation,
14  wherein the native code generation process optimizes the callback by the native
15  code method into the virtual machine, <u>wherein optimizing the interactions</u>
16  <u>involves optimizing calls from the application to the native code method by using</u>
17  <u>additional information from the integrated intermediate representation to reduce</u>
18  <u>the number of indirect calls and indirect references associated with the</u>
19  <u>call</u><s>swherein optimizing the interactions involves using contextual information</s>

7

20 ~~from within the integrated intermediate representation that is generated from the~~
21 ~~native code method as well as the application in order to optimize calls to the~~
22 ~~native code method by the application within the integrated intermediate~~
23 ~~representation.~~

1    29.    (Original) The method of claim 28, wherein the native code
2 generation process also optimizes calls to the native code method by the
3 application.

1    30.    (Original) The method of claim 28, wherein optimizing the
2 callback by the native code method into the virtual machine involves optimizing a
3 callback that accesses a heap object within the virtual machine.

1    31.    (Previously presented) The method of claim 28,
2        wherein the virtual machine is a platform-independent virtual machine;
3 and
4        wherein combining the intermediate representation for the native code
5 method with the intermediate representation associated with the application
6 running on the virtual machine involves integrating calls provided by an interface
7 for accessing native code into the native code method.

1    32.    (Currently amended) A computer-readable storage device storing
2 instructions that when executed by a computer cause the computer to perform a
3 method for reducing the overhead involved in executing native code methods in
4 an application running on a virtual machine, the method comprising:
5        deciding to optimize a callback by any native code method into the virtual
6 machine;

8

7    decompiling at least part of the selected native code method into an
8    intermediate representation;
9    obtaining an intermediate representation associated with the application
10   running on the virtual machine which interacts with the native code method;
11   integrating the intermediate representation for the native code method into
12   the intermediate representation associated with the application running on the
13   virtual machine to form an integrated intermediate representation; and
14   generating native code from the combined intermediate representation,
15   wherein the native code generation process optimizes the callback by the native
16   code method into the virtual machine, wherein optimizing the interactions
17   involves optimizing calls from the application to the native code method by using
18   additional information from the integrated intermediate representation to reduce
19   the number of indirect calls and indirect references associated with the calls
20   ~~wherein optimizing the interactions involves using contextual information from~~
21   ~~within the integrated intermediate representation that is generated from the native~~
22   ~~code method as well as the application in order to optimize calls to the native~~
23   ~~code method by the application within the integrated intermediate representation.~~

1    33.    (Previously presented) The computer-readable storage device of
2    claim 32, wherein the native code generation process also optimizes calls to the
3    native code method by the application.

1    34.    (Previously presented) The computer-readable storage device of
2    claim 32, wherein optimizing the callback by the native code method into the
3    virtual machine involves optimizing a callback that accesses a heap object within
4    the virtual machine.

9

1    35.    (Previously presented) The computer-readable storage device of
2 claim 32,
3        wherein the virtual machine is a platform-independent virtual machine;
4 and
5        wherein combining the intermediate representation for the native code
6 method with the intermediate representation associated with the application
7 running on the virtual machine involves integrating calls provided by an interface
8 for accessing native code into the native code method.

1    36-39. (Canceled)